Quiz Tool

Final Report for CS39440 Major Project

Author: Mr. Michal Goly (mwg2@aber.ac.uk)
Supervisor: Mr. Chris Loftus (cwl@aber.ac.uk)

26th April 2017
Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a BEng degree in Software Engineering (G600)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name: Michal Goly

Date: 26th April 2018

Consent to share this work

By including my name below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name: Michal Goly

Date: 26th April 2018

Acknowledgements

I am grateful to coffee.

Abstract

Student engagement and live knowledge monitoring are vital in the provision of good quality lecture content in 2018. It is important to make sure audience understands concepts presented, and quizzes can help lecturers judge students' understanding in real time.

Aberystwyth University currently uses Qwizdom[1] live polling tool during the provision of some lectures and practical sessions. The university operates under a single license forcing lecturers to book sessions before they can use the tool. Due to human nature, session hijacking occasionally occurs to the bemusement of both students and lecturers. For example, students could be shown biology slides half way through their geography lecture.

This project focused on the design and development of an in-house built Quiz Tool, enabling multiple lecturers to use it at the same time and potentially making Qwizdom redundant in the future. This ambition could only be achieved if the project was of high quality and its future maintainability was considered at all stages of the design and development.

The Quiz Tool allows lecturers to login using their Google Single Sign-on[2] credentials, upload their PDF lecture slides and create *Lectures* in the system. Each *Lecture* can be then edited, and eligible slides can be marked as quizes. True/false, single and multi choice style quizes are supported. Once a lecturer is happy with their *Lecture*, he can broadcast it and receive a session key which can be shared with students. Lecture slides will be shown to all students and the lecture can be delivered in a traditional fashion up to the moment a slide has been marked as a quiz. Students will then be able to answer the question and polling results will be presented in real time to the lecturer. Lecture sessions broadcasted in the past are kept, and students' answers can be exported as a PDF report for future analysis.

The tool is composed of a back-end with an associated database, and two front-ends. One for lecturers and one for students. Finally, the tool has been successfully developed using an agile methodology, adjusted for a single person project.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background & Objectives

1.1 Background

1.1.1 Motivation

I enjoy programming, and this project seemed to involve a lot of coding. I was also excited to build a complex, real time system, while applying my previous software engineering experience and learning new technologies at the same time. I knew I would be given a lot of freedom to choose the most appropriate tools for the task, and incorporate modern software development practices, including continuous integration and containerised environments, to deliver good quality software. I was also keen on developing something that could be actually useful to the university once I leave Aberystwyth. It is more motivating to develop a product, while knowing it could be potentially used in real life, as opposed to being forgotten after the submission. I have experienced being presented with wrong slides during a lecture in my first year at university, so I was happy to address the problem of a single session Qwizdom license with my tool.

1.1.2 Technology Considerations

1.1.2.1 Programming Languages

The initial proposal was to create the classroom quiz system using Java[3] to run it natively on Android[4] mobile devices. The lecturer was supposed to create quizzes on his teaching machine, by interacting with the system using a web front end. Lecture slides were then supposed to be broadcasted to his audience, and they could use their mobile phones to answers questions, which would be then sent back to the lecturer for analysis. I have had previous experience with native Android development, therefore this approach seemed like a reasonable option. The only problem was, iOS[5] devices are very popular in the United Kingdom, and developing an app for Android would exclude a good percentage of students from being able to actively participate in lectures presented. I have therefore started to think about alternative approaches.

The second possibility was to use React Native[6], a JavaScript[7] framework allowing developers to create mobile applications in JavaScript and compile it down to both iOS and Android. This approach would still require the web front end for the lecturer to be developed, and a natural choice would be to use React[8] to keep the learning curve as low as possible.

The final alternative considered, was to develop the whole tool as a web application. This way both the front end for lecturers and students could be developed using the same framework. Members of the audience could participate in lectures by accessing the web application using web browsers installed both on their mobile phones, regardless of the operating system, and their laptops. I considered both React and Angular 4[9], since I have already briefly used it before. React is a library for developing user interface, whereas Angular is a web development framework. The only caveat with using Angular is that the developer needs to learn TypeScript[10], which compiles down to JavaScript.

1.1.2.2 The WebSocket Protocol

The Quiz Tool was supposed to allow a lecturer to broadcast his slides to all the students participating in a lecture. This requires a bidirectional communication protocol between the server and the clients. For example, if a lecturer emits the next slide of his presentation, this change should be pushed to the back end, and then the back end needs to be able to send the new slide to all the students. Students' clients should also be able to push quiz answers back to the back end, so they can be presented to the lecturer in some form.

The WebSocket Protocol can be used to create such real time systems. It enables two-way communication between a client and a remote host, making it ideal for instant messaging, gaming applications, and the Quiz Tool. It uses a single TCP connection for trafic in both directions[11].

1.1.2.3 Prototyping

I have followed various tutorials to quickly prototype proof of concept applications, in order to learn more about the technologies which could be useful during the Quiz Tool development. I started with the Socket.io chat tutorial. Socket.io is a JavaScript library enabling bidirectional event-based communication, and uses the WebSocket Protocol internally[12]. I have created a chat application following one of the tutorials on their homepage[13]. I have also learned the basics of Angular by following the Tour of Heroes tutorial[14], and finally tried to understand how Angular could work together with Socket.io by following the MEAN Socket.io tutorial[15].

1.1.3 On-site vs External Hosting

The initial plan to handle authentication in Quiz Tool was to use the university LDAP[16]. This way, lecturers would be able to provide their university credentials, which would be checked using the Bind operation against the university directory of users, to check if a given person is authorised to use the tool. This would require the Quiz Tool to be deployed to a production environment running within the university intranet. The alternative was to deploy the application to an external cloud hosting.

1.1.4 Similar Tools

Qwizdom[1], is the tool currently used by the university to embed quizzes into presentations to judge students' understanding of the content presented. It has to be installed on lecturer's machine as it is a desktop application. Qwizdom integrates with Microsoft PowerPoint[17], by adding an extra toolbar allowing the lecturer to insert quiz questions into his presentations and then broadcast them. Once a presentation is started, a session key appears which can be shared with the audience. They can then use the key to join the lecture and answer questions once they become available.

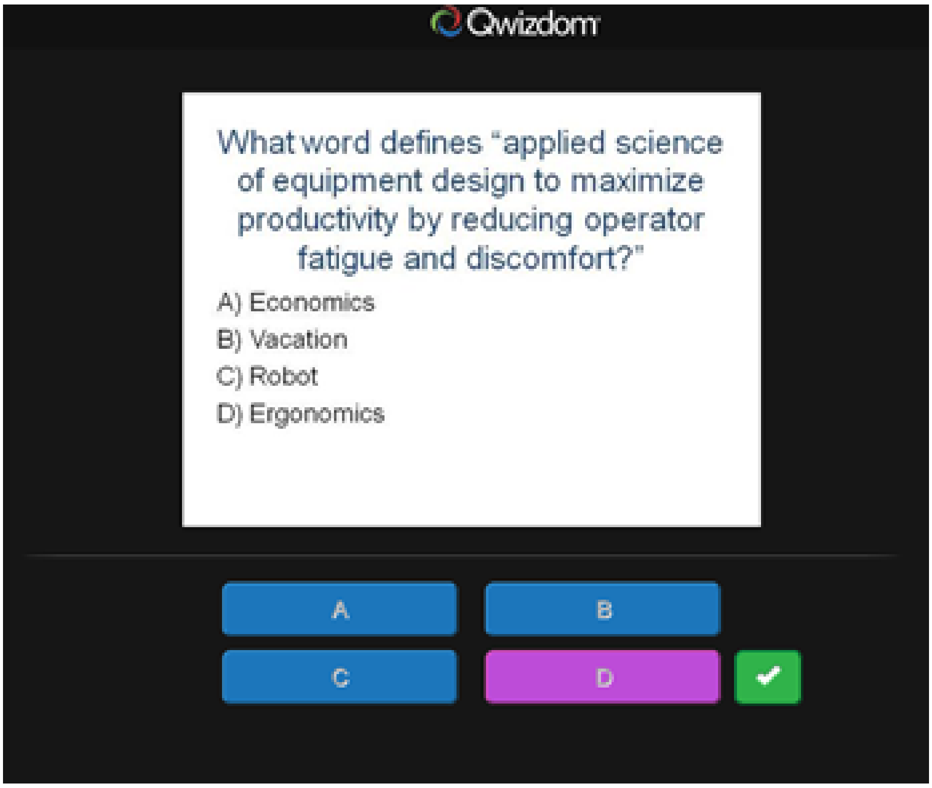

Figure 1.1: Qwizdom web view for the audience

1.2 Analysis

1.2.1 MEAN Stack

Having considered various methods of developing the Quiz Tool, I have decided to choose the web application approach. Allowing students to access the tool using both their mobile devices and laptops, combined with the relatively low learning curve both for me, and for anyone maintaining the software in the future, made it the best option in my opinion. Furthermore, I have decided to develop the application entirely in JavaScript based technologies using the MEAN stack[18].

MEAN stack consists of four elements:

- **M**ongoDB is a JSON document storage NoSQL database
- **E**xpress is a minimalistic JavaScript web development framework
- **A**ngular is a front end web development framework
- **N**odeJS is a JavaScript engine

Front ends for both lecturers and students would be written in Angular, NodeJS would be used on the back end together with Express, and the persistence layer would be provided by MongoDB.

1.2.2 Docker

Each part of the application would be containerised using Docker[19], and containers would run together in the same fashion on the developer's machine, and the build engine using docker-compose[20] container orchestration tool. Docker is an open source engine which can be used to wrap an application and all its dependencies into a lightweight container that can run on any machine capable of running containers[21]. Docker-compose on the other hand, is a tool for defining and running multi-container Docker applications.

1.2.3 AWS Production Environment

Unfortunately, the on-site hosting provided by the university to students to deploy their final projects to was inadequate for the DevOps infrastructure I wanted to put in place. I wanted to have a machine capable of running docker-compose, and some form of continuous integration agent. The LXC debian containers[22] were not compatible with either docker-compose, or Jenkins[23]. The Quiz Tool would be therefore deployed to the production environment provided by an external cloud provider AWS[24], and an alternative build agent would be used. The GitHub Student Developer Pack[25] offers $150 worth of credits for the Amazon cloud, which is why AWS was chosen.

1.2.4 Build

Continuous integration and deployment would be provided by Circle CI[26], as it is relatively easy to work with and is very similar to Travis[27], while not being overly complex like Jenkins[23]. Its major advantage over its competitors is the ability to build projects stored in private repositories, free of charge to a certain amount of build hours per month. Continuous integration and deployment are software development industry practices that enable teams to reliably release new features and products. Code is integrated and merged frequently, which shortens the release cycle, improves code quality and team's productivity[28].

Build of the Quiz Tool should at the very least:

- Checkout the source code from the version control
- Run all the tests
- Deploy the tool to the production environment when a production build runs

1.2.5 Authentication and Security

LDAP authentication could not be used with the production environment provided by AWS. The alternative would be to use the Google Single Sign On[2], where lecturers could login to the tool using their Google credentials. The major drawback of this approach is that anyone with a Google account can login into the tool as a lecturer. The major advantage is, that the credentials handling would be outsourced to Google, making the Quiz Tool more secure and the tool would not have to store hashed passwords by simply relying on the confirmation of user's identity by Google.

Having said that, Google ids would still be stored in MongoDB to uniquely identify lecturers in the future, and to know who is the owner of lectures in the system. Due to the fact that the interaction of the Docker containers making up the Quiz Tool would be managed by docker-compose, the MongoDB container could be hidden from the outside world and could be only accessed from the server container running the back end code. This would make the database more secure.

1.2.6 Top Level Requirements

Even though the tool would be developed using an agile approach, as described in the next section of this document, certain top level functional requirements have been identified before the development work had started. The initial conversation with the client (the supervisor of the project) yielded the following requirements:

- **FR-1** - Add a login page for lecturers. As a lecturer it should be possible to login into the tool using lecturer's credentials.
- **FR-2** - Add a login page for students. As a student it should be possible to join an ongoing session using the session code provided by the lecturer.
- **FR-3** - Add a dashboard for lecturers to upload their lecture slides to.
- **FR-4** - Add the ability to add quizzes to lectures.
- **FR-5** - Add the ability to broadcast lectures.
- **FR-6** - Add the ability to export lecture session results in some format for future analysis.

1.3 Process

1.3.1 Version Control

Git[29] would be used for the version control during the development of the Quiz Tool, together with the GitHub[30] web-based hosting service. The source code would be stored in a private repository. There would be a `master` production branch, and suitable guards would be added so that new code cannot be pushed directly to the production branch. The feature-branch git workflow[31] would be used and each `feature-branch` would need to have an associated issue (story) on GitHub. Then a pull request could be opened between the `feature-branch` and the `master` branch, and Circle CI would checkout the code, run all tests and report back to GitHub to either allow the merge or block it, depending on whether the build was successful or not. Once the pull request is merged into `master`, a production build would run, where Circle CI would checkout the code, run all the tests and finally automatically deploy the new version of the Quiz Tool to the production environment provided by AWS.

A Kanban board provided by the ZenHub Chrome plugin[32] would be used. It would allow issues to be grouped into epics, and each story could have a corresponding amount of points assigned to it, depending on the likely complexity of the task. For example a single point would correspond to half day of work.

1.3.2 Development Methodology

The application would be developed using the SCRUM methodology adjusted for a single person project. The development work would be split into weekly sprints. Each sprint would start with sprint planning, where issues would be created and their complexity would be estimated using ZenHub story points. During the week, issues would be tackled, and the progress could be monitored using the Kanban board. GitHub labels would be used to quickly differentiate between different types of issues. For example there would be a different coloured label for front end tasks, back end tasks, documentation, DevOps etc. Time commitment would also be tracked on day to day basis using a Google Sheets spreadsheet[33]. Each sprint would end with a sprint retrospective, where a document would be produced containing a list of things that went well during the week, and things that could be improved. Weekly meetings with the client (supervisor) would provide immediate feedback on work done and allow re-adjustment of the approach necessary to stay on track to deliver the software before the deadline. Finally, velocity would be tracked and burn down charts automatically produced following each iteration. This would give more confidence in estimating the future work.

Chapter 2

Design & Implementation

The structure of this chapter follows the agile methodology used to develop the Quiz Tool. Each section contains information about a sprint, allowing the reader to gain more understanding of how the design and the tool itself evolved over time.

2.1 Sprint 1 - Hello Quiz Tool

2.1.1 Sprint Planning

The first sprint of the Quiz Tool focused on setting up the DevOps of the project, and deployment of a "hello world" version of the tool consisting of the front end, back end and nginx[34] running together using docker-compose. It was also important to investigate how to best structure the application to include both the front and the back end of the application in a single GitHub repository. The following subsections cover the most important aspects of the sprint, and the entire list of estimated stories can be found in the Appendix C of this report.

2.1.2 Application Structure

The main goal of using docker-compose, was to have the whole application running in the same manner locally on the developer's machine, during testing on Circle CI, and in the production environment. This meant the application had to be containerised using Docker, and containers had to be able to communicate with each other appropriately. This was even more difficult considering Socket.io had to be incorporated, to allow real time broadcast of lecture slides to students in the future. I have decided to create a prototype of a very basic chat application, containerised using Docker and orchestrated using docker-compose. The prototype had to be written in the MEAN stack, and use Socket.io to prove it was possible to make all technologies work together.

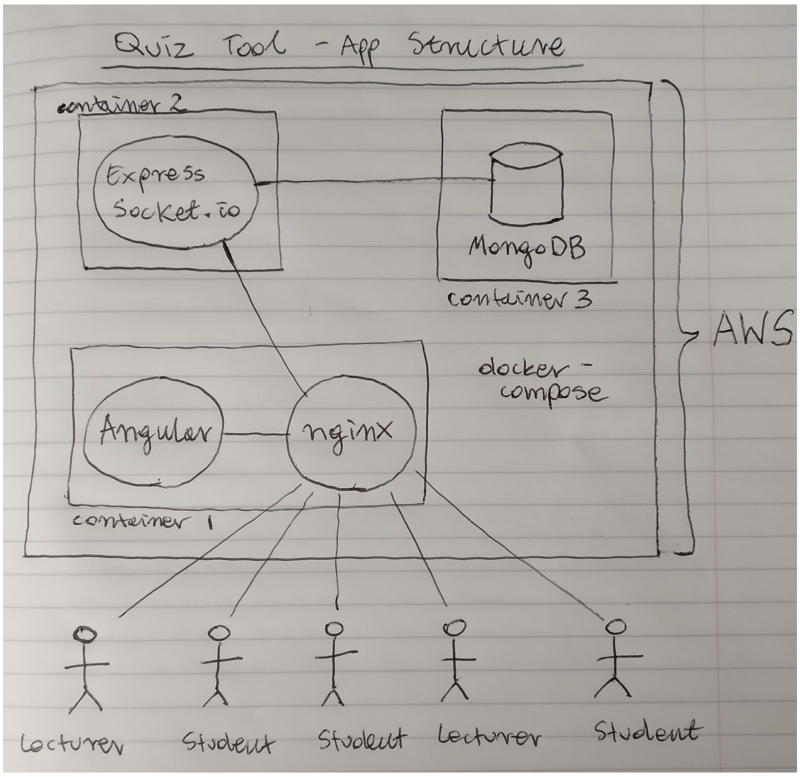

Figure 2.1: The proof of concept application structure

```
version: '2.0'
services:
  client:
    build: client
    ports:
      - "80:80"
    links:
      - server_node
  server_node:
    build: server
    links:
      - database
  database:
    image: mongo
    ports:
      - "27017:27017"
```

Figure 2.2: The docker-compose.yml file describing the tool's structure

2.1.2.1 Front End Container

The front end container consisted of Angular 4 and nginx reverse proxy. The structure has been based on the *Dockerized Angular 4 App (with Angular CLI)* repository[35]. The Socket.io client dependency has been added to allow sending messages to the back end using sockets. The initial `Dockerfile` included in the Appendix E of this report as the Figure E.1 uses the multi-stage build added in Docker 17.05. The Angular app is compiled to JavaScript and HTML files during the initial stage of the build, and then these files are copied to the nginx public folder to be served to clients. This results in a lean, production ready image.

2.1.2.2 Back End Container

The back end container included in the Appendix E of this report as the Figure E.2 consisted of a Node.js runtime, the Express framework and the Socket.io engine capable of pushing messages to clients using Sockets. The `Dockerfile` illustrates the very basic Node container.

2.1.2.3 Database Container

Finally, the MongoDB Docker image has been pulled automatically from the official mongo Docker Hub registry[36].

2.1.3 Continuous Integration

Circle CI has been integrated with the GitHub repository containing the source code of the Quiz Tool. Every time a pull request was made, Circle CI would be notified. It would then assign a virtual machine build agent from a pool, and spin up a clean build environment. It would then checkout the code and run the steps specified in the build config file, before reporting if the build was successfull back to GitHub.

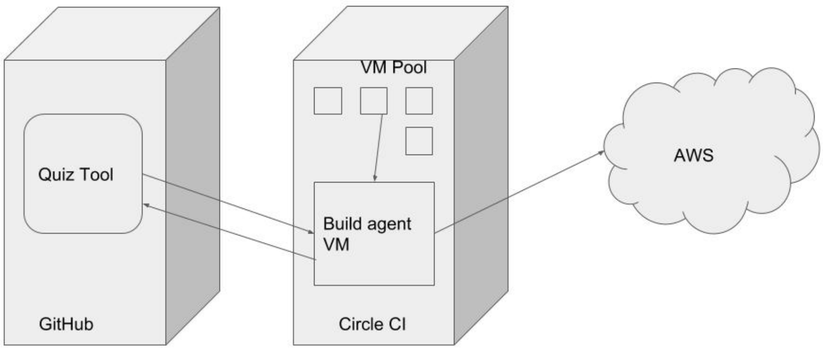

Figure 2.3: Continuous Integration and Deployment

The `.circleci/config.yml` file included in the Appendix E of this report as the Figure E.3, describes the initial build steps of the Quiz Tool. Code is checked out from the version control, all the dependencies necessary to perform following steps are installed, the project is then built and started using docker-compose, before the `curl localhost` command checks if the application is up and running. Finally, if the current branch being built is `master`, the `deploy.sh` bash script runs, which deploys the application to production.

2.1.4 Production Environment

The production environment of the tool is hosted on the AWS cloud. The AWS Elastic Beanstalk has been chosen specificaly, as applications in various programming languages can be deployed with ease, wihout having to worry about the infrastructure running these applications[37]. The Multicontainer Docker AWS Elastic Beanstalk[38] environment, creates a single Amazon EC2[39] instance and uses ECS (Amazon Elastic Container Service)[40] to coordinate container deployments to multicontainer Docker environments. The similarity with docker-compose, means the Quiz Tool would behave similarly on developer's machine, in testing and in production.

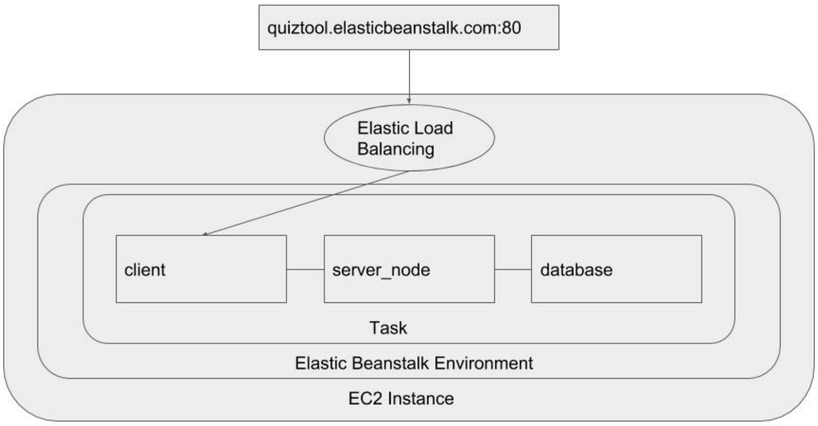

Figure 2.4: Quiz Tool Production Environment

`docker-compose.yml` files define how docker-compose should run Docker containers together. `Dockerrun.aws.json` is the equivalent configuration file to specify relationships between Docker containers running in the Multicontainer Elastic Beanstalk environment. The format of the config file included in the Appendix E of this report as the Figure E.4 is very similar to the docker-compose configuration files. The major difference is that the AWS Elastic Beanstalk does not build Docker images itself, and images have to be pulled dynamically from Docker registries. Docker registries are simply servers used for storage and distribution of Docker images.

2.1.4.1 Circle CI and AWS Integration

The build agent automatically deploys the tool to production when the `master` branch is being built. The bash script included in the Appendix E of this report as the Figure E.5 performs the actual deployment, and the approach is based on the examples[41][42]. The `$AWS_ACCESS_KEY_ID` and the `$AWS_SECRET_ACCESS_KEY` environment variables have been added to the build configuration using the Circle CI web panel. The integration has been achieved by creating an AWS profile config file and installing the `awsebcli` Elastic Beanstalk command line utility as one of the build steps. Both `client` and `server_node` containers are then tagged and pushed to the private Docker image registry provided by Amazon Elastic Container Service, before the `eb deploy prod-env` command actually triggers the production deployment. Both front and back end containers are then pulled from the private registry specified in the `Dockerrun.aws.json` file, while the mongo image is pulled directly from the official mongo Docker hub.

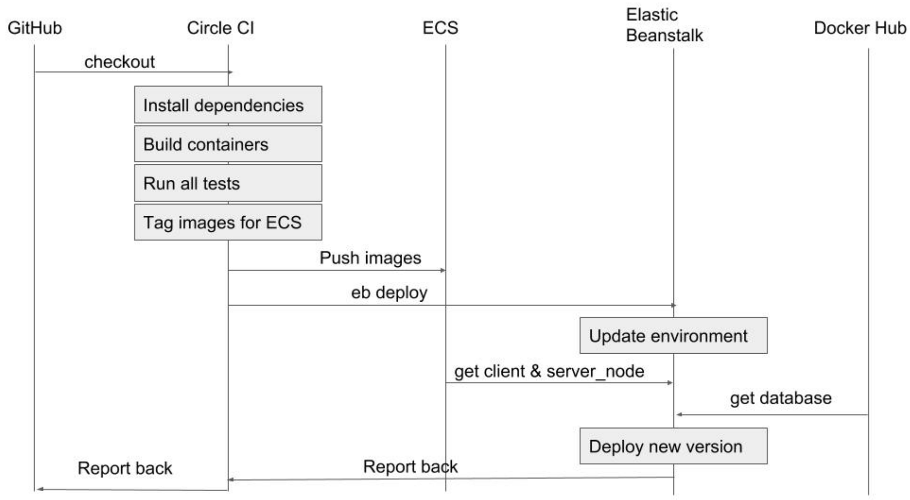

Figure 2.5: Production Deployment Visualised

2.1.5 Story Boards

Including the customer early in the development and design process allows teams to stay on track and readjust their design if necessary. The story boards below have been produced to gain feedback from the client, and ease the front end development in future iterations.

Figure 2.6 shows how the lectuer would login into the tool using Google Single Sign On. He could then upload his lecture slides using an action button in the bottom right corner of the screen, be able to edit his slides and mark certain slides as quizzes. A list of cards would be presented showing all lectures belonging to the lecturer and he could broadcast them to his audience. Subsequently, he could navigate through the slides and slides with embedded quizzes would split the screen in half to show a bar chart with answers as they come in. Finally, lecture would end once he clicks the end button.

Figure 2.7 on the other hand, shows how students could interact with the tool. Student could join an ongoing lecture using a session key provided by the lecturer. She could then participate in the lecture by watching the slides, and submitting answers to quizzes as they come in. The correct answer could be presented back to her in a form of a bar chart.

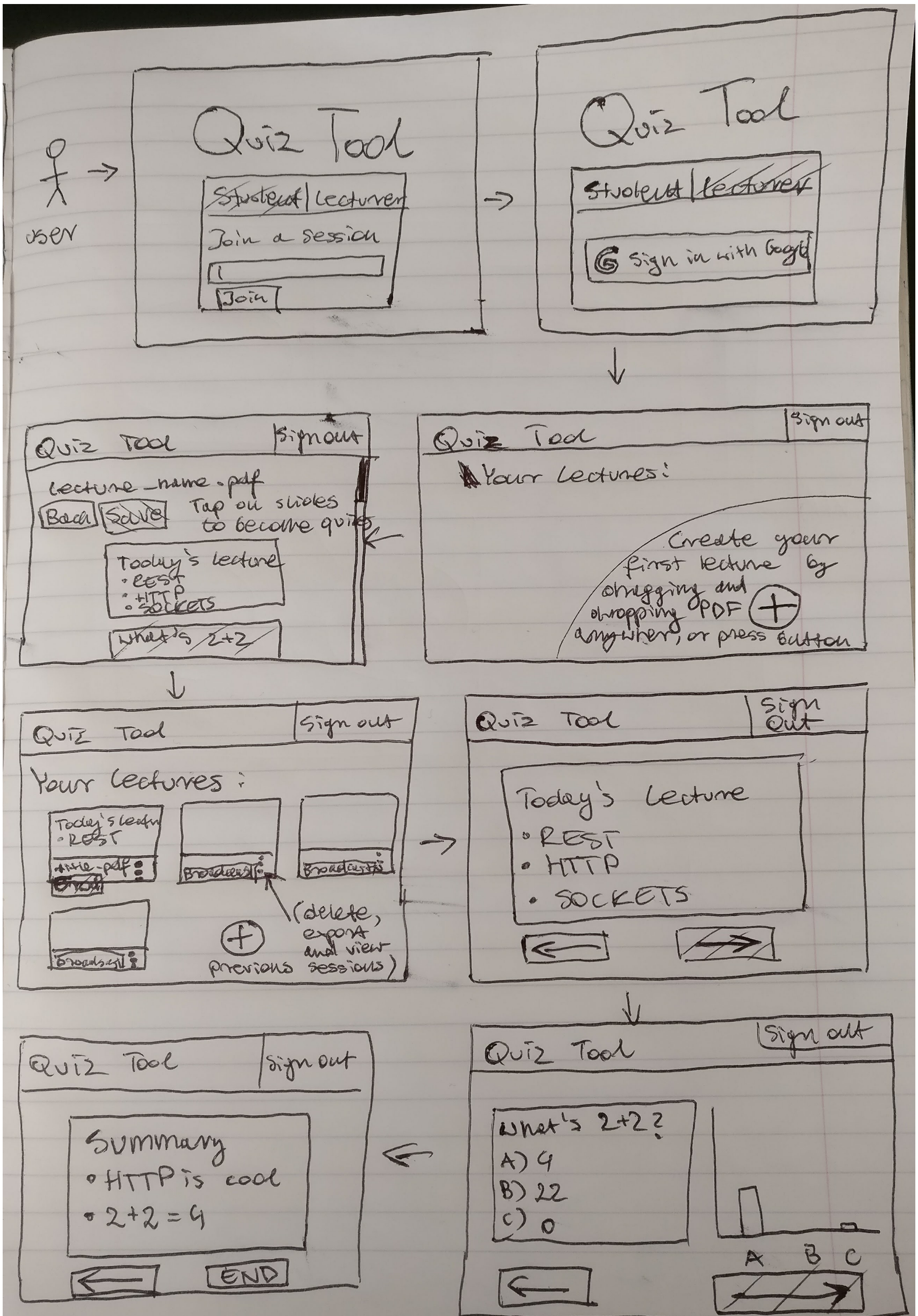

Figure 2.6: Story Board Lecturer Interaction

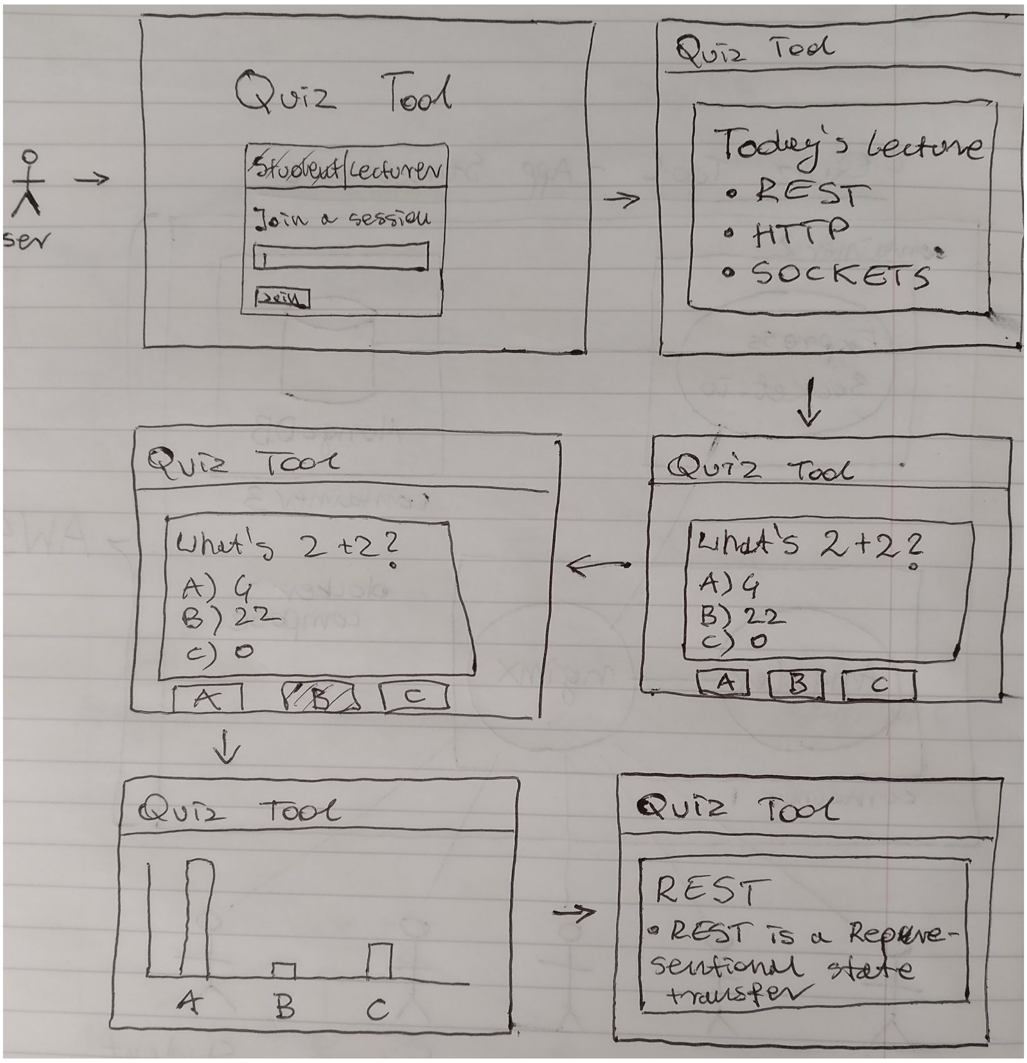

Figure 2.7: Story Board Student Interaction

2.1.6 Sprint Retrospective

The first sprint proved it was possible to deploy a MEAN stack, containerised application to production using Circle CI. Crucial DevOps has been successfully put in place, and the initial feedback has been gathered from the client thanks to the low fidelity prototypes. The full sprint retrospective document produced can be found in the Appendix D of this report.

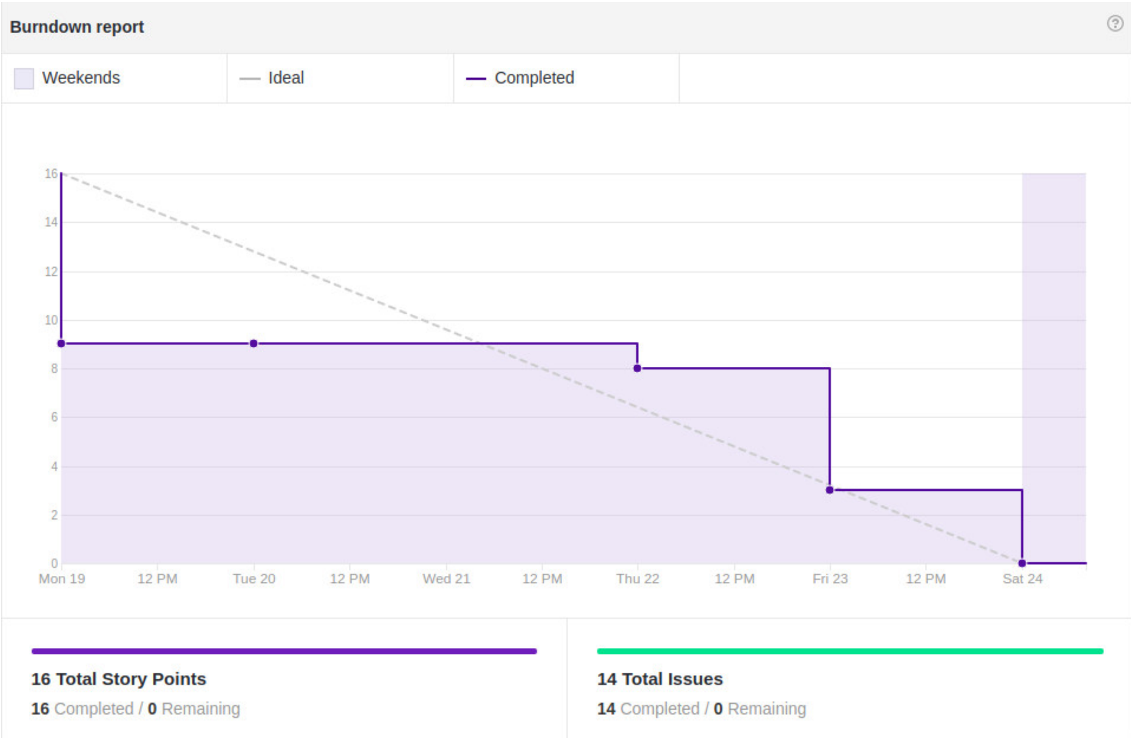

Figure 2.8: Burndown Chart Sprint 1

2.2 Sprint 2 - Bare Bone Application

2.2.1 Sprint Planning

This sprint focused on converting the proof of concept chat application created in the previous week and converting it into a basic Quiz Tool. Simple login page for both students and lecturers would be developed based on the paper prototypes from the previous sprint and Google Single Sign On would be added. Once logged in, lecturers could upload their PDF lecture slides and broadcast them to all students, as session generation was not implemented yet. The entire list of estimated stories can be found in the Appendix C of this report.

2.2.2 Back End Application Structure

2.2.3 Front End Application Structure

2.2.4 Login Page and Authorisation

- JWT tokens - JWT interceptor - Google single sign on - session cookies

2.2.5 Persistence Layer

- mongoose

2.2.6 Lecture Upload

2.2.7 Lecture Broadcast

2.2.8 Sprint Retrospective

2.3 Sprint 3 - Add Quizes

2.3.1 Sprint Planning

This iteration focused on the ability to create quizzes, embed them into slides and broadcast only to people who have the appropriate session key. The entire list of estimated stories can be found in the Appendix C of this report.

2.3.2 Sprint Retrospective

2.4 Sprint 4 - Fancy Quizes and Defect Fixing

2.4.1 Sprint Planning

This sprint focused on addressing a critial bug in the production environment concerning the PDF to PNG conversion. Extra types of quizzes would also be added to complement the single choice ones availble at the moment. The entire list of estimated stories can be found in the Appendix C of this report.

2.4.2 Sprint Retrospective

2.5 Sprint 5 - Persistence, Report Generation and Testing

2.5.1 Sprint Planning

The final sprint focused on the persistence of students' answers in the database and generation of PDF reports based on these answers. User friendly error handler has also been added, and the application has been polished before the Quiz Tool implementation was over. As always, the entire list of estimated stories can be found in the Appendix C of this report.

2.5.2 Sprint Retrospective

Chapter 3

Final Design

This chapter follows on the discussion of the individual sprints and summarises the final design of the tool.

Chapter 4

Testing

4.1 Server Side Unit Tests

4.2 Client Side Unit Tests

4.3 Selenium Integration Tests

Chapter 5

Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Other questions can be addressed as appropriate for a project.

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

In the latter stages of the module, we will discuss the evaluation. That will probably be around week 9, although that differs each year.

Appendix A

Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. If third party code or libraries are used, your work will build on that to produce notable new work. The key requirement is that we understand what is your original work and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

As an example, you might include a definition such as:

Apache POI library - The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the client's existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [**?**]. The library is released using the Apache License [**?**]. This library was used without modification.

Appendix B

Ethics Submission

Ethics Application Number: 9563

AU Status
Undergraduate or PG Taught

Your aber.ac.uk email address
mwg2@aber.ac.uk

Full Name
Michal Wojciech Goly

Please enter the name of the person responsible for reviewing your assessment.
Prof. Reyer Zwiggelaar

Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment
rrz@aber.ac.uk

Supervisor or Institute Director of Research Department
cs

Module code (Only enter if you have been asked to do so)
CS39440

Proposed Study Title
MMP - Quiz Tool

Proposed Start Date
29/01/2018

Proposed Completion Date
04/05/2018

Are you conducting a quantitative or qualitative research project?
Mixed Methods

Does your research require external ethical approval under the Health Research Authority?
No

Does your research involve animals?
No

Are you completing this form for your own research?
Yes

Does your research involve human participants?
No

Institute
IMPACS

Please provide a brief summary of your project (150 word max)
A web application allowing lecturers to upload their PDF lecture slides, convert certain slides to quizzes (e.g. a slide with bullet points into a single choice A) B) C) quiz), and then broadcasting them to students to monitor their understanding of lecture content presented.

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?
Not applicable

Will appropriate measures be put in place for the secure and confidential storage of data?
Yes

Does the research pose more than minimal and predictable risk to the researcher?
Not applicable

Will you be travelling, as a foreign national, in to any areas that the UK Foreign and Commonwealth Office advise against travel to?
No

Please include any further relevant information for this section here:

If you are to be working alone with vulnerable people or children, you may need a DBS (CRB) check. Tick to confirm that you will ensure you comply with this requirement should you identify that you require one.
Yes

Declaration: Please tick to confirm that you have completed this form to the best of your knowledge and that you will inform your department should the proposal significantly change.
Yes

Please include any further relevant information for this section here:

Appendix C

Sprint Stories

This appendix contains the user stories produced before each sprint.

3.1 Sprint 1

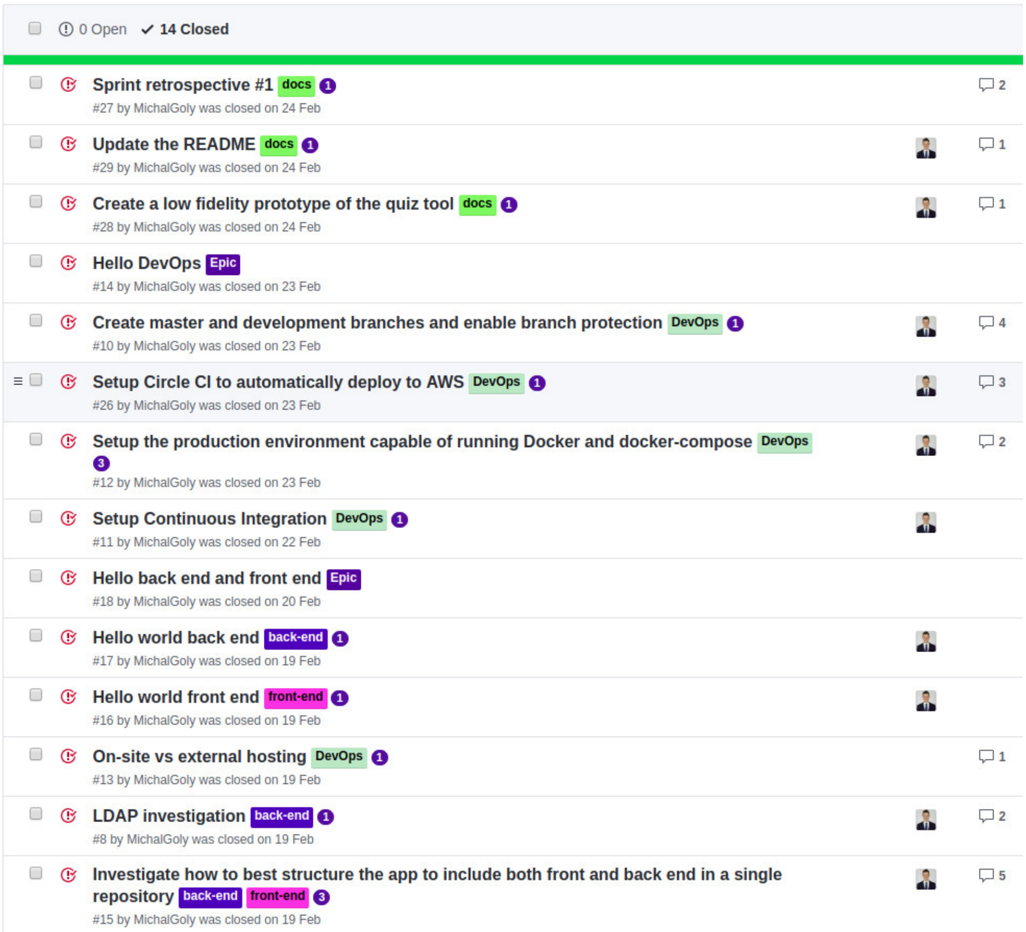

Figure C.1: Stories Sprint 1

3.2 Sprint 2

3.3 Sprint 3

3.4 Sprint 4

3.5 Sprint 5

Appendix D

Sprint Retrospective Documents

This appendix contains the sprint retrospective documents produced at the end of each sprint.

Quiz Tool - Sprint #1 Retrospective

What went well:

- LDAP authentication blocker resolved by going with the Google Single Sign On for Lecturers authentication
- Creation of the proof of concept “hello world” front and back ends capable on running in a docker-compose network with an nginx reverse proxy and socket.io
- DevOps setup is now complete. There is a Circle CI build in place which runs builds each time there is a pull request and auto deploys to AWS if master branch is being built.
- Circle CI <-> AWS integration. The proof of concept app is now running in production on the Amazon Multicontainer Elasticbeanstalk Environment and both the client and server Docker images are automatically pushed and stored in the Amazon Elastic Container Registry.
- I have created low fidelity prototypes of the tool which will allow me to plan the development in the future iterations
- I kept track of the time spent using google sheets

What could be improved:

- I have only put 20 hours into the Final Project work this week due to 2 job interviews
- I could have probably squized more issues into the sprint, but I decided to start a new sprint one day early instead.
- Some stories estimated to take half-day work took 10 minutes, whereas others had to be extended to a day or a day and a half. Points allocation accuracy should hopefully improve once actual coding begins in the following sprints.

Appendix E

Code Examples

This appendix contains code examples relevant to the discussion in the main body of the report.

5.1 Intial Front End Dockerfile

The initial `Dockerfile` of the client container added in the first sprint of the development of the tool.

```
### STAGE 1: Build ###

# We label our stage as 'builder'
FROM node:8-alpine as builder

COPY package.json package-lock.json ./

RUN npm set progress=false && npm config set depth 0 && npm cache
    clean --force

## Storing node modules on a separate layer will prevent
    unnecessary npm installs at each build
RUN npm i && mkdir /ng-app && cp -R ./node_modules ./ng-app

WORKDIR /ng-app

COPY . .

## Build the angular app in production mode and store the artifacts
     in dist folder
RUN $(npm bin)/ng build --prod --build-optimizer

### STAGE 2: Setup ###

FROM nginx:1.13.3-alpine

## Copy our default nginx config
COPY nginx/default.conf /etc/nginx/conf.d/

## Remove default nginx website
RUN rm -rf /usr/share/nginx/html/*

## From 'builder' stage copy over the artifacts in dist folder to
    default nginx public folder
COPY --from=builder /ng-app/dist /usr/share/nginx/html

CMD ["nginx", "-g", "daemon off;"]
```

Figure E.1: Intial Front End Dockerfile

5.2 Initial Back End Dockerfile

The initial `Dockerfile` of the server_node container added in the first sprint of the development of the tool.

```
FROM node:carbon
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD [ "npm", "start" ]
```

Figure E.2: Back End Dockerfile

5.3 Intial Circle CI Config File

The initial `.circleci/config.yml` added in the first sprint of the development.

```
version: 2
jobs:
  build:
    machine: true

    working_directory: ~/repo

    steps:
      - checkout

      - run:
          name: install docker-compose
          command: |
            set -x
            sudo chown -R $(whoami) /usr/local/bin
            curl -L https://github.com/docker/compose/releases/
                download/1.11.2/docker-compose-`uname -s`-`uname -m`
                > /usr/local/bin/docker-compose
            chmod +x /usr/local/bin/docker-compose

      - run:
          name: docker compose build image
          command: |
            set -x
            docker-compose build
            docker-compose up

      - run:
          name: unit tests
          command: |
            curl localhost

      # deploy only master branch
      - deploy:
          command: |
            if [ "${CIRCLE_BRANCH}" == "master" ]; then
              chmod +x scripts/deploy.sh
              ./scripts/deploy.sh
            fi
```

Figure E.3: Initial Build Config File

5.4 Initial Dockerrun.aws.json file

The initial version of the file specifying how containers should be linked together when running on the production environment on AWS.

```
{
    "AWSEBDockerrunVersion": 2,
    "containerDefinitions": [
        {
            "name": "client",
            "image": "993389244112.dkr.ecr.eu-west-2.amazonaws.com/
                quiz-tool-client:latest",
            "memory": 128,
            "essential": true,
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80
                }
            ],
            "links": [
                "server_node"
            ]
        },
        {
            "name": "server_node",
            "image": "993389244112.dkr.ecr.eu-west-2.amazonaws.com/
                quiz-tool-server:latest",
            "memory": 128,
            "essential": true,
            "links": [
              "database"
            ]
        },
        {
          "name": "database",
          "image": "mongo",
          "memory": 128,
          "essential": true,
          "portMappings": [
              {
                  "hostPort": 27017,
                  "containerPort": 27017
              }
          ]
        }
    ]
}
```

Figure E.4: Dockerrun.aws.json

5.5 Production Deployment Script

The bash script performing the deployment from Circle CI to the production environment hosted on AWS.

```
#!/bin/bash
set -x
set -e

AWS_CONFIG_FILE=$HOME/.aws/config

mkdir $HOME/.aws
touch $AWS_CONFIG_FILE
chmod 600 $AWS_CONFIG_FILE

echo "[default]"                                        >
    $AWS_CONFIG_FILE
echo "aws_access_key_id=$AWS_ACCESS_KEY_ID"             >>
    $AWS_CONFIG_FILE
echo "aws_secret_access_key=$AWS_SECRET_ACCESS_KEY"     >>
    $AWS_CONFIG_FILE

$(aws ecr get-login --no-include-email --region eu-west-2)

docker tag repo_client:latest 993389244112.dkr.ecr.eu-west-2.
    amazonaws.com/quiz-tool-client:latest
docker tag repo_server_node:latest 993389244112.dkr.ecr.eu-west-2.
    amazonaws.com/quiz-tool-server:latest

docker push 993389244112.dkr.ecr.eu-west-2.amazonaws.com/quiz-tool-
    client:latest
docker push 993389244112.dkr.ecr.eu-west-2.amazonaws.com/quiz-tool-
    server:latest

eb deploy prod-env
```

Figure E.5: Production Deployment Script

Bibliography

[1] Qwizdom, Qwizdom homepage, 2018. [Online] Available: https://qwizdom.com/, [Accessed: Apr. 9, 2018].

Qwizdom is the tool currently used by the univeristy and will be replaced with this project.

[2] Google Sign-In, ”Google Identity homepage”, 2018. [Online] Available: https://developers.google.com/identity/, [Accessed: Apr. 10, 2018].

Google Sign-In is a secure authentication system that reduces the burden of login for your users, by enabling them to sign in with their Google account. The same account they already use with Gmail, Play, Google+, and other Google services.

[3] Java Programming Language, ”Java homepage”, 2018. [Online] Available: https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html, [Accessed: Apr. 11, 2018].

The Java Programming Language is a general-purpose, concurrent, strongly typed, class-based object-oriented language. It is normally compiled to the bytecode instruction set and binary format defined in the Java Virtual Machine Specification.

[4] Android, ”Android homepage”, 2018. [Online] Available: https://www.android.com/, [Accessed: Apr. 11, 2018].

Android is a mobile operating system developed by Google.

[5] iOS, ”Apple homepage”, 2018. [Online] Available: https://www.apple.com/uk/ios/ios-11/, [Accessed: Apr. 11, 2018].

iOS is a mobile operating system created and developed by Apple Inc.

[6] React Native, ”React Native homepage”, 2018. [Online] Available: https://facebook.github.io/react-native/, [Accessed: Apr. 11, 2018].

React Native lets you build mobile apps for both iOS and Android using only JavaScript.

[7] JavaScript Programming Language, ”JavaScript Mozilla docs”, 2018. [Online] Available: https://developer.mozilla.org/bm/docs/Web/JavaScript, [Accessed: Apr. 11, 2018].

JavaScript is a lightweight, interpreted, programming language, best known for being the scripting language of the web.

[8] React, ”React homepage”, 2018. [Online] Available: https://reactjs.org/, [Accessed: Apr. 11, 2018].

A JavaScript library for building user interfaces.

[9] Angular, ”Angular homepage”, 2018. [Online] Available: https://angular.io/, [Accessed: Apr. 11, 2018].

Angular is a TypeScript-based front end development framework supported by Google.

[10] TypeScript Programming Language, ”TypeScript homepage”, 2018. [Online] Available: https://www.typescriptlang.org/, [Accessed: Apr. 11, 2018].

TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.

[11] I. Fette, A. Melnikov, ”The WebSocket Protocol”, [Online] Available: https://tools.ietf.org/html/rfc6455, [Accessed: Apr. 11, 2018].

The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.

[12] Socket.io, ”Socket.io homepage”, 2018. [Online] Available: https://socket.io/, [Accessed: Apr. 11, 2018].

Socket.io is a JavaScript library enabling bidirectional event-based communication.

[13] Chat Application Tutorial, ”Socket.io homepage”, 2018. [Online] Available: https://socket.io/get-started/chat/, [Accessed: Apr. 11, 2018].

A Socket.io tutorial guiding the student in developing a basic chat application using Socket.io.

[14] Tour of Heroes Tutorial, ”Angular homepage”, 2018. [Online] Available: https://angular.io/tutorial, [Accessed: Apr. 11, 2018].

The Tour of Heroes tutorial covers the fundamentals of Angular, by building an app that helps staffing agency manage its stable of heroes.

[15] Building Chat Application using MEAN Stack (Angular 4) and Socket.io, ”djamware.com”, 2018. [Online] Available: https://www.djamware.com/post/58e0d15280aca75cdc948e4e/building-chat-application-using-mean-stack-angular-4-and-socketio, [Accessed: Apr. 11, 2018].

Step by step tutorial of building simple chat application using MEAN stack (Angular 4) and Socket.io.

[16] J. Sermersheim, Ed. Novell, ”Lightweight Directory Access Protocol (LDAP): The Protocol”, [Online] Available: https://tools.ietf.org/html/rfc4511, [Accessed: Apr. 11, 2018].

Aberystwyth University directory of users could be checked using the Bind operation to authenticate lecturers once they provide their email and password.

[17] Microsoft PowerPoint, ”products.office.com”, 2018. [Online] Available: https://products.office.com/en-gb/powerpoint, [Accessed: Apr. 11, 2018].

A proprietary tool owned by Microsoft to create presentations.

[18] MEAN stack, [Online] Available: https://github.com/linnovate/mean, [Accessed: Apr. 11, 2018].

The MEAN stack uses Mongo, Express, Angular(4) and Node for simple and scalable full-stack js applications.

[19] Docker, ”Docker homepage”, [Online] Available: https://www.docker.com/, [Accessed: Apr. 11, 2018].

Docker allows app to be containerised and run on multiple hosts in the same fashion.

[20] docker-compose, ”docker-compose homepage”, [Online] Available: https://docs.docker.com/compose/, [Accessed: Apr. 11, 2018].

Compose is a tool for defining and running multi-container Docker applications.

[21] M. OǴara, ”Ben Golub, Who Sold Gluster to Red Hat, Now Running dotCloud”, 26th July 2013. [Online] Available: http://maureenogara.sys-con.com/node/2747331, [Accessed: Apr. 11, 2018].

A journal article mentioning what Docker does.

[22] LXC, ”LXC”, [Online] Available: https://wiki.debian.org/LXC, [Accessed: Apr. 11, 2018].

Linux Containers (LXC) provide a Free Software virtualization system for computers running GNU/Linux. They are inappropriate for the task at hand, as they do not allow running docker-compose due to the lack of system permissions.

[23] Jenkins, ”Jenkins homepage”, [Online] Available: https://jenkins.io/, [Accessed: Apr. 11, 2018].

Jenkins is an industry proven automation tool. It is typically deployed on premises as a Java web application and gives developers more control over the build agents compared to its competitors.

[24] Amazon Web Services, ”AWS homepage”, [Online] Available: https://aws.amazon.com/, [Accessed: Apr. 11, 2018].

AWS provides cloud computing plaftoms to individuals, companies and governments. The production environment of Quiz Tool is hosted on AWS.

[25] GitHub, ”Student Developer Pack”, [Online] Available: https://education.github.com/pack, [Accessed: Apr. 11, 2018].

GitHub partners with top companies to allow students gain experience with real world tools. It funded the production environment hosting on AWS of the Quiz Tool.

[26] Circle CI, ”Circle CI homepage”, [Online] Available: https://circleci.com/, [Accessed: Apr. 12, 2018].

Circle CI is very similar to Travis, but it allows a certain amount of free build hours per month for private repositories.

[27] Travis, ”Travis homepage”, [Online] Available: https://travis-ci.org/, [Accessed: Apr. 12, 2018].

Travis is a popular continuos integration tool hosted typically on an external cloud. Free to use for open source projects.

[28] M. Shahin, M. Ali Babar, and L. Zhu, ”Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices”, 2017, [Online] Available: https://arxiv.org/pdf/1703.07019.pdf, [Accessed: Apr. 12, 2018].

This paper describes what continuos integration and delivery are.

[29] Git, Git homepage, 2018, [Online] Available: https://git-scm.com/, [Accessed: Apr. 12, 2018].

Version control system used for the Quiz Tool development.

[30] GitHub, Inc., GitHub homepage, 2018, [Online] Available: http://github.com/, [Accessed: Apr. 12, 2018].

A popular version control web hosting offering private repositories to students. Code will be stored there and checkout by CI for testing and deployment.

[31] Git Feature Branch Workflow, [Online] Available: https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow, [Accessed: Apr. 12, 2018].

Each issue has an associated feature branch, and code necessary to develop the issue is commited to the appropriate feature branch. Then a pull request is made between the feature branch and development/master, and once all tests pass the feature branch can be merged to development/master.

[32] ZenHub, ”ZenHub homepage”, [Online] Available: https://www.zenhub.com/, [Accessed: Apr. 12, 2018].

ZenHub is an agile project management chrome extension for GitHub. It provides the ability to assign points to issues, create epics, provides velocity tracing burndown charts for milestones, and a Kanban board to better manage work at hand.

[33] Google Sheets, ”Google docs homepage”, [Online] Available: https://www.google.co.uk/sheets/about/, [Accessed: Apr. 12, 2018].

A SaaS tool for creating spreadsheets owned by Google. A spreadsheet has been used to track time commitment on day to day basis during the development of the Quiz Tool.

[34] nginx, ”nginx homepage”, [Online] Available: https://www.nginx.com/, [Accessed: Apr. 12, 2018].

Nginx is a web server which is used as a reverse proxy in the Quiz Tool.

[35] avatsaev, ”Dockerized Angular 4 App (with Angular CLI)”, [Online] Available: https://github.com/avatsaev/angular4-docker-example, [Accessed: Apr. 12, 2018].

The starting point of the client side of the Quiz Tool.

[36] mongo, ”mongo official docker repository”, [Online] Available: https://hub.docker.com/_/mongo/, [Accessed: Apr. 12, 2018].

The official docker hub repository of MongoDB. The image is used by the Quiz Tool in both production and development.

[37] What Is AWS Elastic Beanstalk, ”AWS docs”, [Online] Available: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html, [Accessed: Apr. 13, 2018].

The Quiz Tool is deployed to AWS Elastic Beanstalk production environment.

[38] Multicontainer Docker Environments, ”AWS docs”, [Online] Available: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create_deploy_docker_ecs.html, [Accessed: Apr. 13, 2018].

The Multicontainer AWS Elastic Beanstalk is what actually orchestrates Docker containers in production.

[39] Amazon EC2, ”EC2 homepage”, [Online] Available: https://aws.amazon.com/ec2/, [Accessed: Apr. 13, 2018].

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

[40] Amazon ECS, ”ECS homepage”, [Online] Available: https://aws.amazon.com/ecs/, [Accessed: Apr. 13, 2018].

Amazon Elastic Container Service stores Docker images of the Quiz Tool tagged and ready for deployment into production.

[41] How to setup Elastic Beanstalk Deployment, ”circleci forum”, [Online] Available: https://discuss.circleci.com/t/how-to-setup-elastic-beanstalk-deployment/6154/4, [Accessed: Apr. 13, 2018].

The forum post shows an example of an AWS deployment script which was adapted for Quiz Tool deployment.

[42] Settings to deploy to AWS Elastic Beanstalk on CircleCi (EB Cli 3) , ”gist”, [Online] Available: https://gist.github.com/RobertoSchneiders/9e0e73e836a80d53a21e, [Accessed: Apr. 13, 2018].

The gist post shows an example of an AWS deployment script which was adapted for Quiz Tool deployment.